PORTAL
USPTO

Subscribe (Full Service) Register (Limited Service, Free) Login

Search: ● The ACM Digital Library ○ The Guide

force effect, cache, priority, full, allocation SEARCH

Feedback Report a problem Satisfaction survey

Terms used **force effect cache priority full allocation** Found **57,686** of **185,178**

Sort results by: relevance
Display results: expanded form

Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next
Best 200 shown
Relevance scale

1 GPGPU: general purpose computation on graphics hardware
David Luebke, Mark Harris, Jens Krüger, Tim Purcell, Naga Govindaraju, Ian Buck, Cliff Woolley, Aaron Lefohn
August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes SIGGRAPH '04**
Publisher: ACM Press
Full text available: pdf(63.03 MB) Additional Information: full citation, abstract

The graphics processor (GPU) on today's commodity video cards has evolved into an extremely powerful and flexible processor. The latest graphics architectures provide tremendous memory bandwidth and computational horsepower, with fully programmable vertex and pixel processing units that support vector operations up to full IEEE floating point precision. High level languages have emerged for graphics hardware, making this computational power accessible. Architecturally, GPUs are highly parallel s ...

2 The implications of cache affinity on processor scheduling for multiprogrammed, shared memory multiprocessors
Raj Vaswani, John Zahorjan
September 1991 **ACM SIGOPS Operating Systems Review , Proceedings of the thirteenth ACM symposium on Operating systems principles SOSP '91**, Volume 25 Issue 5
Publisher: ACM Press
Full text available: pdf(1.57 MB) Additional Information: full citation, abstract, references, citings, index terms

In a shared memory multiprocessor with caches, executing tasks develop "affinity" to processors by filling their caches with data and instructions during execution. A scheduling policy that ignores this affinity may waste processing power by causing excessive cache refilling.Our work focuses on quantifying the effect of processor reallocation on the performance of various parallel applications multiprogrammed on a shared memory multiprocessor, and on evaluating how the magnitude of this cost aff ...

3 Computing curricula 2001
September 2001 **Journal on Educational Resources in Computing (JERIC)**
Publisher: ACM Press
Full text available: pdf(613.63 KB) html(2.78 KB) Additional Information: full citation, references, citings, index terms

4 Cache Memories
Alan Jay Smith
September 1982 **ACM Computing Surveys (CSUR)**, Volume 14 Issue 3
Publisher: ACM Press
Full text available: pdf(4.61 MB) Additional Information: full citation, references, citings, index terms

5 A dynamic processor allocation policy for multiprogrammed shared-memory multiprocessors
Cathy McCann, Raj Vaswani, John Zahorjan
May 1993 **ACM Transactions on Computer Systems (TOCS)**, Volume 11 Issue 2
Publisher: ACM Press
Full text available: pdf(2.26 MB) Additional Information: full citation, abstract, references, citings, index terms

We propose and evaluate empirically the performance of a dynamic processor-scheduling policy for multiprogrammed shared-memory multiprocessors. The policy is dynamic in that it reallocates processors from one parallel job to another based on the currently realized parallelism of those jobs. The policy is suitable for implementation in production systems in that: —It interacts well with very efficient user-level thread packages, leaving to them many low-level thr ...

Keywords: shared memory parallel processors, threads, two-level scheduling

6 External memory algorithms and data structures: dealing with **massive data**
Jeffrey Scott Vitter
June 2001 **ACM Computing Surveys (CSUR)**, Volume 33 Issue 2
Publisher: ACM Press
Full text available: pdf(828.46 KB) Additional Information: full citation, abstract, references, citings, index terms

Data sets in large applications are often too massive to fit completely inside the computers internal memory. The resulting input/output communication (or I/O) between fast internal memory and slower external memory (such as disks) can be a major performance bottleneck. In this article we survey the state of the art in the design and analysis of external memory (or EM) algorithms and data structures, where the goal is to exploit locality in order to reduce the I/O costs. We consider a varie ...

Keywords: B-tree, I/O, batched, block, disk, dynamic, extendible hashing, external memory, hierarchical memory, multidimensional access methods, multilevel memory, online, out-of-core, secondary storage, sorting

7 The priority-based coloring approach to register allocation
Fred C. Chow, John L. Hennessy
October 1990 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 12 Issue 4
Publisher: ACM Press
Full text available: pdf(2.97 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Global register allocation plays a major role in determining the efficacy of an optimizing compiler. Graph coloring has been used as the central paradigm for register allocation in modern compilers. A straightforward coloring approach can suffer from several shortcomings. These shortcomings are addressed in this paper by coloring the graph using a priority ordering. A natural method for dealing with the spilling emerges from this approach. The detailed algorithms for a priority-based colori ...

8 Real-time shading

Marc Olano, Kurt Akeley, John C. Hart, Wolfgang Heidrich, Michael McCool, Jason L. Mitchell, Randi Rost

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(7.39 MB) Additional Information: full citation, abstract

Real-time procedural shading was once seen as a distant dream. When the first version of this course was offered four years ago, real-time shading was possible, but only with one-of-a-kind hardware or by combining the effects of tens to hundreds of rendering passes. Today, almost every new computer comes with graphics hardware capable of interactively executing shaders of thousands to tens of thousands of instructions. This course has been redesigned to address today's real-time shading capabili ...

9 Level set and PDE methods for computer graphics

David Breen, Ron Fedkiw, Ken Museth, Stanley Osher, Guillermo Sapiro, Ross Whitaker

August 2004 **Proceedings of the conference on SIGGRAPH 2004 course notes SIGGRAPH '04**

Publisher: ACM Press

Full text available: pdf(17.07 MB) Additional Information: full citation, abstract, citings

Level set methods, an important class of partial differential equation (PDE) methods, define dynamic surfaces implicitly as the level set (iso-surface) of a sampled, evolving nD function. The course begins with preparatory material that introduces the concept of using partial differential equations to solve problems in computer graphics, geometric modeling and computer vision. This will include the structure and behavior of several different types of differential equations, e.g. the level set eq ...

10 A parallel, incremental, mostly concurrent garbage collector for servers

Katherine Barabash, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Yoav Ossia, Avi Owshanko, Erez Petrank

November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6

Publisher: ACM Press

Full text available: pdf(683.50 KB) Additional Information: full citation, abstract, references, index terms

Multithreaded applications with multigigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multigigabyte heap while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multicycle fence instructions required by weak ordering to a minimum.We designed and implemented a collector facing these demands building on th ...

Keywords: Garbage collection, JVM, concurrent garbage collection

11 Scheduler activations: effective kernel support for the user-level management of parallelism

Thomas E. Anderson, Brian N. Bershad, Edward D. Lazowska, Henry M. Levy

February 1992 **ACM Transactions on Computer Systems (TOCS)**, Volume 10 Issue 1

Publisher: ACM Press

Full text available: pdf(2.04 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Threads are the vehicle for concurrency in many approaches to parallel programming. Threads can be supported either by the operating system kernel or by user-level library code in the application address space, but neither approach has been fully satisfactory. This paper addresses this dilemma. First, we argue that the performance of kernel threads is inherently worse than that of user-level threads, rather than this being an artifact of existing ...

Keywords: multiprocessor, thread

12 A simple interprocedural register allocation algorithm and its effectiveness for LISP
Peter A. Steenkiste, John L. Hennessy
January 1989 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 11 Issue 1
Publisher: ACM Press

Full text available: pdf(2.56 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Register allocation is an important optimization in many compilers, but with per-procedure register allocation, it is often not possible to make good use of a large register set. Procedure calls limit the improvement from global register allocation, since they force variables allocated to registers to be saved and restored. This limitation is more pronounced in LISP programs due to the higher frequency of procedure calls. An interprocedural register allocation algorithm is developed by simp ...

13 Streamlining data cache access with fast address calculation
Todd M. Austin, Dionisios N. Pnevmatikatos, Gurindar S. Sohi
May 1995 **ACM SIGARCH Computer Architecture News , Proceedings of the 22nd annual international symposium on Computer architecture ISCA '95**, Volume 23 Issue 2
Publisher: ACM Press

Full text available: pdf(1.58 MB) Additional Information: full citation, abstract, references, citings, index terms

For many programs, especially integer codes, untolerated load instruction latencies account for a significant portion of total execution time. In this paper, we present the design and evaluation of a fast address generation mechanism capable of eliminating the delays caused by effective address calculation for many loads and stores.Our approach works by predicting early in the pipeline (part of) the effective address of a memory access and using this predicted address to speculatively access the ...

14 Query evaluation techniques for large databases
Goetz Graefe
June 1993 **ACM Computing Surveys (CSUR)**, Volume 25 Issue 2
Publisher: ACM Press

Full text available: pdf(9.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Database management systems will continue to manage large data volumes. Thus, efficient algorithms for accessing and manipulating large sets and sequences will be required to provide acceptable performance. The advent of object-oriented and extensible database systems will not solve this problem. On the contrary, modern data models exacerbate the problem: In order to manipulate large sets of complex objects as efficiently as today's database systems manipulate simple records, query-processi ...

Keywords: complex query evaluation plans, dynamic query evaluation plans, extensible database systems, iterators, object-oriented database systems, operator model of parallelization, parallel algorithms, relational database systems, set-matching algorithms, sort-hash duality

15 Avoidance and suppression of compensation code in a trace scheduling compiler
Stefan M. Freudenberger, Thomas R. Gross, P. Geoffrey Lowney
July 1994 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 16 Issue 4
Publisher: ACM Press
Full text available: pdf(3.58 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Trace scheduling is an optimization technique that selects a sequence of basic blocks as a trace and schedules the operations from the trace together. If an operation is moved across basic block boundaries, one or more compensation copies may be required in the off-trace code. This article discusses the generation of compensation code in a trace scheduling compiler and presents techniques for limiting the amount of compensation code: avoidance (restricting code motion so that no compensatio ...

Keywords: SPEC89, instruction-level parallelism, performance evaluation, trace scheduling

16 Scheduler activations: effective kernel support for the user-level management of parallelism
Thomas E. Anderson, Brian N. Bershad, Edward D. Lazowska, Henry M. Levy
September 1991 **ACM SIGOPS Operating Systems Review , Proceedings of the thirteenth ACM symposium on Operating systems principles SOSP '91**, Volume 25 Issue 5
Publisher: ACM Press
Full text available: pdf(1.68 MB) Additional Information: full citation, abstract, references, citings, index terms

Threads are the vehicle for concurrency in many approaches to parallel programming. Threads separate the notion of a sequential execution stream from the other aspects of traditional UNIX-like processes, such as address spaces and I/O descriptors. The objective of this separation is to make the expression and control of parallelism sufficiently cheap that the programmer or compiler can exploit even fine-grained parallelism with acceptable overhead.Threads can be supported either by the op ...

17 Eliminating synchronization bottlenecks using adaptive replication
Martin C. Rinard, Pedro C. Diniz
May 2003 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 25 Issue 3
Publisher: ACM Press
Full text available: pdf(826.28 KB) Additional Information: full citation, abstract, references, index terms, review

This article presents a new technique, adaptive replication, for automatically eliminating synchronization bottlenecks in multithreaded programs that perform atomic operations on objects. Synchronization bottlenecks occur when multiple threads attempt to concurrently update the same object. It is often possible to eliminate synchronization bottlenecks by replicating objects. Each thread can then update its own local replica without synchronization and without interacting with other threads. When ...

Keywords: Atomic operations, commutativity analysis, parallel computing, parallelizing compilers, replication, synchronization

18 4.2BSD and 4.3BSD as examples of the UNIX system
John S. Quarterman, Abraham Silberschatz, James L. Peterson
December 1985 **ACM Computing Surveys (CSUR)**, Volume 17 Issue 4
Publisher: ACM Press
Full text available: pdf(4.07 MB) Additional Information: full citation, abstract, references, citings, index terms, review

This paper presents an in-depth examination of the 4.2 Berkeley Software Distribution, Virtual VAX-11 Version (4.2BSD), which is a version of the UNIX Time-Sharing System. There are notes throughout on 4.3BSD, the forthcoming system from the University of California at Berkeley. We trace the historical development of the UNIX system from its conception in 1969 until today, and describe the design principles that have guided this development. We then present the internal data structures and ...

19 Empirical performance evaluation of concurrency and coherency control protocols for database sharing systems
Erhard Rahm
June 1993 **ACM Transactions on Database Systems (TODS)**, Volume 18 Issue 2
Publisher: ACM Press
Full text available: pdf(3.37 MB) Additional Information: full citation, abstract, references, citings, index terms, review

Database Sharing (DB-sharing) refers to a general approach for building a distributed high performance transaction system. The nodes of a DB-sharing system are locally coupled via a high-speed interconnect and share a common database at the disk level. This is also known as a "shared disk" approach. We compare database sharing with the database partitioning (shared nothing) approach and discuss the functional DBMS components that require new and coordinated solutions for DB-shar ...

Keywords: coherency control, concurrency control, database partitioning, database sharing, performance analysis, shared disk, shared nothing, trace-driven simulation

20 Idletime scheduling with preemption intervals
Lars Eggert, Joseph D. Touch
October 2005 **ACM SIGOPS Operating Systems Review , Proceedings of the twentieth ACM symposium on Operating systems principles SOSP '05**, Volume 39 Issue 5
Publisher: ACM Press
Full text available: pdf(2.02 MB) Additional Information: full citation, abstract, references, index terms

This paper presents the idletime scheduler; a generic, kernel-level mechanism for using idle resource capacity in the background without slowing down concurrent foreground use. Many operating systems fail to support transparent background use and concurrent foreground performance can decrease by 50% or more. The idletime scheduler minimizes this interference by partially relaxing the work conservation principle during *preemption intervals*, during which it serves no background requests eve ...

Keywords: background processing, disk scheduler, idletime scheduling, network scheduler, preemption interval, queuing, resource scheduler

Results 1 - 20 of 200 Result page: **1** 2 3 4 5 6 7 8 9 10 next

EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	106069	cache	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:29
L2	5447	force adj effect	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:15
L3	969	memory adj (presentation or representation)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:15
L4	550559	priority	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:16
L5	19443	memory near3 full$4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:17
L6	42	1 and 2	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:17
L7	252	3 and 4	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:17
L8	5	6 and 7	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:17
L9	5	5 and 8	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:20

EAST Search History

L10	75966	(sufficient or enough) near2 (memory or space)	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L11	108	2 and 10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L12	25	1 and 11	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L13	25	4 and 12	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L14	1254768	full	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L15	25	13 and 14	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:22
L16	18323496	@ad<"19990922"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:51
L17	12	15 and 16	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:28
L18	30794	"711"/$.ccls.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:28
L19	2	15 and 18	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:28

EAST Search History

L20	4223	cache near3 allocat$5	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:29
L21	8	3 and 20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:31
L22	1	10 and 21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:30
L23	1	16 and 21	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:32
L24	19	2 and 20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:31
L25	11	16 and 24	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:35
L26	5	2 and 3	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:35
L27	1	16 and 26	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:35
L28	108	2 and 10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:35
L29	68	16 and 28	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:36

EAST Search History

L30	0	18 and 29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:36
L31	11	20 and 29	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:38
L32	8	3 and 20	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:44
L33	18325832	16 adn 32	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:39
L34	1	16 and 32	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:39
L35	2	"6252383".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:45
L36	2	"6252583".pn.	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:45
L37	17815138	@ad<"19990505"	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:51
L38	617	20 and 10	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:52
L39	204	37 and 38	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:52

EAST Search History

L40	103	18 and 39	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:53
L41	20	5 and 40	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:53
L42	0	3 and 41	US-PGPUB; USPAT; EPO; JPO; DERWENT; IBM_TDB	OR	OFF	2006/09/17 00:53